# CS 370: Software Design & Development

- Class Modules
    - Software Terminology
    - Software Methodology
    - Software Patterns and Architecture
    - Practical Application
- Supporting Coursework
    - Labs: Platform Proficiency
- Resources

## Class Modules

### Software Terminology

Time is spent discussing common terms and general concepts related to software design and abstract architecture (in-depth patterns and architecture covered in a subsequent module). The principle behind this module is the establishment of a common set of terms and concepts that the instructor and the students can refer to when investigating and discussing more advanced software topics later in the course.

Suggested Topics: Definition of Software Terms, Multi-Tier Architecture, Serialization/Deserialization (process and formats), DRY, SoC, Abstraction Layers

### Software Methodology

Time is spent discussing past and contemporary approaches to developing software from a methodological perspective, including topics on organization of work, process for defining and prioritizing requirements, and defining visual designs and information architecture. The principle behind this module is to establish a framework for students to execute the course requirements in the Practical Application module. Design components often presented by guest lecturer in the field.

Suggested Topics: Waterfall, Agile, Scrum (Roles and Processes), Definition of Requirements, Conceptualization of Software Implementation Verticals

### Software Patterns and Architecture

Time is spent discussing classic and contemporary software implementation patterns and architectures (including OOP), as well as good development tooling and practices such github/gitflow, unit testing and release/versioning management. The principle behind this module is the establishment of good practices for the execution of the Practical Application module, and the base assumption is that most students will have had little-or-no exposure to OOP tenets and patterns.

Suggested Topics: Principles of OOP, Abstract Classes, Interfaces, Builder Pattern, Factory Pattern, Singleton Pattern, Internal Classes and Scoping, Model-View-Controller, Model-View-Presenter, Model-View-ViewModel, Unit Testing, Git Commands and Flow

### Practical Application

Students are divided into teams of 3 - 5 and required to implement a software application of their own imagining over the second half of the semester (roughly 8 weeks). Course subject matter covered in subsequent modules will be applied to the Practical Application module. The weekly Practical Application cycle has two separate components:

- **Project Steering**
  Instructor meets with each group to assess progress and provide guidance if the group is experiencing issues with implementation (whether social, process or technical).

    - **Project Demo**
      All student teams present weekly progress to class. Questions and feedback from instructor and peers used to help refine the product.

## Supporting Coursework

### Labs: Platform Proficiency

Students may not have had an opportunity to work on a contemporary software platform. Lab work on a single platform is provided in the first half of the semester to provide enough of a grounding for students to pursue the implementation of the Practical Application module on that platform. It is expected that students will pursue platform solutions independently if they are not specifically covered during class time.

## Resources

- Android Studio Download
- Android Developer Documentation